funcction verify-sentence

The definition of predicate calculus sentences and the examples just presented suggest a method for verifying that an expression is a sentence. This is written as a recursive algorithm, ***verify_sentence, verify_sentence*** takes as argument a candidate expression and return success if the expression is a sentence.

```
Function verify-sentence (expression);
begin
    case
        expression is an atomic sentence: return SUCCESS;
        expression is of the form Q X s, where Q is either ∀ or ∃, X is a variable,
          if verify-sentence(s) returns SUCCESS then return SUCCESS
            else return FAIL;
        expression is of the form ¬s:
          if verify-sentence(s) returns SUCCESS then return SUCCESS
            else return FAIL;
        expression is of the form s1 op s2 where op is a binary logical operator:
          if verify-sentence (s1) returns SUCCESS and verity-sentence (s2) returns SUCCESS
              then return SUCCESS
          else return FAIL;
        otherwise: return FAIL
        end case
end.
```

We conclude this section with an example of the use of predicate calculus to describe a simple world. The domain of discourse is a set of family relationships in a biblical genealogy:

mother(marry, john)
mother(marry, susie)
father(adam, john)
father(adam, susie)
∀X ∀Y father(X, Y) ∨ mother(X, Y) → parent(X, Y)
∀X ∀Y ∀Z parent(X, Y) ∧ parent(X, Z) → sibling(Y, Z)

In this example we use the predicates **mother** and **father** to define a set of parent-child relationships. The implications give general definitions of other relationships, such as parent and sibling, in terms of these predicates. Intuitively, it is clear that these implications can be used to infer facts such as **sibling(susie,john)**. To formalize this process so that it can be performed on a computer, care must be taken to define inference algorithms and to ensure that such algorithms indeed draw correct conclusions from a set of predicate calculus

assertions.

2.2.2 A Semantics for the Predicate Calculus

Having defined well-formed expressions in the predicate calculus, it is important to determine their meaning in terms of objects, properties, and relations in the world. Predicate calculus semantics provide a formal basis for determining the truth value of well-formed expressions. The truth of expressions depends on the mapping of constants, variables, predicates, and functions into objects and relations in the domain of discourse. The truth of relationships in the domain determines the truth of the corresponding expressions.

For example: information about a person, **George**, and his friends **Kate** and **Susie** may be expressed by:

friends(george, susie)

friends(george, kate)

If it is indeed true that **George** is a friend of **Susie** and **George** is a friend of **Kate** then these expressions would each have the truth value (assignment) **T**. If **George** is a friend of **Susie** but not of **Kate**, then the first expression would have truth value **T** and the second would have truth value **F**. To use the predicate calculus as a representation for problem solving, we describe objects and relations in the domain of interpretation with a set of well-formed expressions. The terms and predicates of these expressions denote objects and relations in the domain. This database of predicate calculus expressions, each having truth value **T**, describes the "state of the world." The description of **George** and his friends is a simple example of such a database.

DEFINITIONS: INTERPRETATION

Let the domain **D** be a nonempty set.

An ***interpretation*** over **D** is an assignment of the entities of **D** to each of the constant, variable, predicate, and function symbols of a predicate calculus expression, such that:

1. Each constant is assigned an element of **D**.
2. Each variable is assigned to a nonempty subset of **D**; these are the allowable substitutions for that variable.
3. Each **function f** of **arity m** is defined on m arguments of **D** and defines a mapping from $\mathbf{D}^m$ into **D**.
4. Each **predicate p** of **arity n** is defined on n arguments from **D** and defines a mapping from $\mathbf{D}^n$ into **{T, F}**.

Given an interpretation, the meaning of an expression is a truth value assignment over the interpretation.

DEFINITIONS: TRUTH VALUE OF PREDICATE CALCULUS EXPRESSIONS

Assume an expression **E** and an interpretation **I** for **E** over a nonempty domain **D**. The truth value for **E** is determined by:

1. The value of a constant is the element of **D** it is assigned to by **I**.
2. The value of a variable is the set of elements of **D** it is assigned to by **I**.
3. The value of a function expression is that element of **D** obtained by evaluating the function for the parameter values assigned by the interpretation.
4. The value of truth symbol "**true**" is **T** and "**false**" is **F**.
5. The value of an atomic sentence is either **T** or **F**, as determined by the interpretation **I**.
6. The value of the negation of a sentence is **T** if the value of the sentence is **F** and is **F** if the value of the sentence is **T**.
7. The value of the conjunction of two sentences is **T** if the value of both sentences is **T** and is **F** otherwise.

8-10. The truth value of expressions using $\vee$, $\rightarrow$, and $\equiv$ is determined from the value of their operands.

***Finally, for a variable X and a sentence S containing X*:**

11. The value of $\forall$ **X S** is **T** if **S** is **T** for all assignments to **X** under **I**, and it is **F** otherwise.
12. The value of $\exists$ **X S** is **T** if there is an assignment to **X** in the interpretation under which S is

T; otherwise it is **F**.

Quantification of variables is an important part of predicate calculus semantics. When a variable appears in a sentence, such as **X** in **likes(george**, **X)**, the variable functions as a placeholder. Any constant allowed under the interpretation can be substituted for it in the expression. Substituting **kate** or **susie** for **X** in **likes(george, X)** forms the statements **likes(george, kate)** and **likes(george, susie).**

The variable **X** stands for all constants that might appear as the second parameter of the sentence. This variable name might be replaced by any other variable name, such as **Y** or PEOPLE, without changing the meaning of the expression. Thus the variable is said to be a *dummy.* In the predicate calculus, variables must be *quantified* in either of two ways: *universally* or *existentially.* A *free variable* is considered if it is not within the scope of either the universal or existential quantifiers. An expression is *closed* if all of its variables are quantified. A *ground expression* has no variables at all. In the predicate calculus, all variables must be quantified.

The symbol indicating universal quantification is ∀. Parentheses are often used to indicate the *scope* of quantification, that is, the instances of a variable name over which quantification holds. Thus **∀X (p(X) ∨ q(Y) → r(X))** indicates that **X** is universally quantified in both **p(X)** and **r(X).** Universal quantification introduces problems in computing the truth value of a sentence, because all the possible values of a variable symbol must be tested to see whether the expression remains true. For example, to test the truth value of **∀X likes(george, X)**, where **X** ranges over the set of all humans, all possible values for **X** must be tested. If the domain of an interpretation is infinite, exhaustive testing of all substitutions to a universally quantified variable is computationally impossible: the algorithm may never halt. Because of this problem, the predicate calculus is said to be ***undecidable.*** Because the propositional calculus does not support variables, sentences can only have a finite number of truth assignments, and we can exhaustively test all these possible assignments. This is done with the truth table. Variables may also be quantified ***existentially.*** In this case the expression

containing one variable is said to be true for at least one substitution from the domain of definition. The existential quantifier is indicated by∃. The scope of an existentially quantified variable is also indicated by enclosing the quantified occurrences of the variable in parentheses. Evaluating the truth of an expression containing an existentially quantified variable may be no easier than evaluating the truth of expressions containing universally quantified variable. Suppose we attempt to determine the truth of the expression by trying substitutions until one is found that makes the expression true. If the domain of the variable is infinite and the expression is false under all substitutions, the algorithm will never halt.

Several relationships between negation and the universal and existential quantifiers are given below. The notion of a variable name as a dummy symbol that stands for a set of constants is also noted. For predicates p and q and variables **X** and **Y**:

$$\neg\exists X\ p(X) \equiv \forall X\ \neg p(X)$$

$$\neg\forall X\ p(X) \equiv \exists X\ \neg p(X)$$

$$\exists X\ p(X) \equiv \exists Y\ p(Y)$$

$$\forall X\ q(X) \equiv \forall Y\ q(Y)$$

$$\forall X\ (p(X) \wedge q(X)) \equiv \forall X\ p(X) \wedge \forall Y\ q(Y)$$

$$\exists X\ (p(X) \vee q(X)) \equiv \exists X\ p(X) \vee \exists Y\ q(Y)$$

In the language we have defined, universally and existentially quantified variables may refer only to objects (constants) in the domain of discourse. Predicate and function names may not be replaced by quantified variables. This language is called the ***first-order predicate calculus.***

DEFINITIONS: FIRST-ORDER PREDICATE CALCULUS

First-order predicate calculus allows quantified variables to refer to objects in the domain of discourse and not to predicates or functions.

For example: ∀ **(Likes) Likes(george, kate) ,** is not a well-formed expression in the first-order predicate calculus. There are *higher-order* predicate calculi where such expressions are meaningful. Many grammatically correct English sentences can be represented in the first-order predicate calculus using the symbols, connectives, and variable symbols defined in this

section. *It is important to note that there is no unique mapping of sentences into predicate calculus expressions;* in fact, an English sentence may have any number of different predicate calculus representations. A major challenge for AI programmers is to find a scheme for using these predicates that optimizes the expressiveness and efficiency of the resulting representation.

Examples: of English sentences represented in predicate calculus are:

- **"If it doesn't rain on Monday, Tom will go to the mountains."**

  $\neg$weather(rain, monday) $\rightarrow$ go(tom, mountains)

- **"Emma is a Doberman Pinscher and a good dog."**

  isa(emma, doberman) $\wedge$ gooddog(emma)

- **"All basketball players are tall."**

  $\forall$X (basketballplayer(X) $\rightarrow$ tall(X))

- **"Some people like anchovies."**

  $\exists$X (person(X) $\wedge$ likes(X, anchovies))

- **"If wishes were horses, beggars would ride."**

  equal(wishes, horses) $\rightarrow$ ride(beggars)

- **"Nobody likes taxes."**

  $\neg\exists$ X likes(X, taxes)